VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2013 **Matěj Marčišovský**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Matěj Marčišovský**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: Poski.com s.r.o.
2. Struktura závěrečné zprávy:
a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta
b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti
c) Zvolený postup řešení zadaných úkolů
d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe
e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe
f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vedl odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Pavel Moravec, Ph.D.**

Konzultant bakalářské práce: Ing. Tomáš Frydrych

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2013

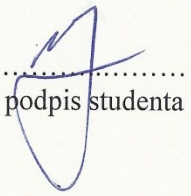
doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne: 7. května 2013



.....
podpis studenta

Poděkování

Rád bych poděkoval Ing. Pavlu Moravcovi, Ph.D. za odbornou pomoc a konzultaci při vytváření této bakalářské práce.

Také bych chtěl poděkovat společnosti Poski.com s.r.o. za poskytnutou příležitost a dobrý kolektiv, zvláště pak Ing. Tomáši Frydrychovi a Ing. Vladimíru Vaňkovi a všem zaměstnancům Poski.com s.r.o. za jejich ochotu a odbornou konzultaci.

Prohlášení zástupce spolupracující právnické nebo fyzické osoby

„Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava."

Dne: 07. 05. 2013

.....
podpis zástupce

Abstrakt

Cílem této bakalářské práce je zdokumentovat a popsat průběh absolvování individuální odborné praxe ve společnosti Poski.com s.r.o. počínaje přijetím a zařazením na pozici Junior Programátora a konče vyhodnocením praxe. Dále dokument obsahuje seznam zadaných úkolů a prací včetně postupu jejich řešení. Součástí tohoto popisu jsou i nabyté zkušenosti a vědomosti, které byly nutné k vypracování konkrétních prací. Závěr shrnuje přínos praxe a praktické uplatnění mé práce.

Abstract

Goal of this thesis is to document and describe process of individual professional practice in the Poski.com s.r.o. company starting with recruitment to the position of Junior Programmer and ending with evaluation of practice. The thesis also describes associated tasks and works including their solution procedures. Further, we describe acquired experience and knowledge which were necessary for development of specific projects. The conclusion of this thesis summarizes benefits of this practice and practical use of the projects.

Klíčová slova

PHP, MySQL, JSON, HTML5, CSS3, JavaScript, ECMA Script, Poski Kecálek, Poski REAL, chat

Keywords

PHP, MySQL, JSON, HTML5, CSS3, JavaScript, ECMA Script, Poski Kecálek, Poski REAL, chat

Seznam použitých zkratek

Zkratka	Anglický význam	Český význam
SVN	Apache Subversion	Apache Subversion
IDE	Integrated Development Environment	Integrované vývojové prostředí
PHP	Hypertext Preprocessor	Hypertextový preprocesor
JSON	JavaScript Object Notation	JavaScriptová objektová notace
XML	Extensible Markup Language	Rozšiřitelný značkovací jazyk
HTML5	HyperText Markup Language 5	Značkovací jazyk pro hypertext 5
CSS3	Cascading Style Sheets 3	Kaskádové styly 3
AJAX	Asynchronous JavaScript and XML	Asynchronní JavaScript a XML

Obsah

1 Úvod 1
2 Poski.com s.r.o. 2
2.1 Seznámení se společností 2
2.2 Popis mého pracovního zařazení 2
2.3 Průběh praxe 3
3 Poski Kecálek 4
3.1 Použité technologie a standardy 4
4 Zvolený postup řešení 6
4.1 Databáze 6
4.2 Jádro systému 10
4.3 Uživatelské rozhraní operátorů a administrátorů 14
5 Uplatněné a chybějící znalosti a dovednosti 19
6 Závěrečné shrnutí 20
7 Použitá literatura 21
Přílohy I

1 Úvod

Pro svou bakalářskou práci jsem si vybral individuální odbornou praxi především proto, abych získal zkušenosti a rozpoznal své reálné schopnosti a možnosti uplatnění. K tomuto účelu jsem si z mnoha společností vybral Poski.com s.r.o., která mne zaujala jak svými referencemi, tak dostupností a popisem nabízené praxe.

Mou hlavní náplní praxe byl vývoj vlastního komunikačního softwaru, který by umožnil komunikaci zákazníků, kteří navštíví daný web, s operátory, kteří jsou k tomuto účelu přihlášení do uživatelského rozhraní softwaru. Jednalo se tedy o komplexní webovou aplikaci. Na programu jsem pracoval sám po celou dobu praxe a občas jsem požádal některého z pracovníků společnosti Poski.com s.r.o. o radu či pomoc.

V dalších kapitolách postupně popisuji společnost, ve které jsem praxi vykonával, průběh praxe, kde popisuji své pracovní zařazení a podrobnější časový rozpis praxe. Následuje popis produktu, nazvaného Poski Kecálek, na kterém jsem pracoval. V kapitole 4 se věnuji řešení zadaného úkolu, které jsem zvolil. V následující kapitole shrnuji schopnosti a dovednosti, které mi scházely a které jsem se naopak naučil už dříve během studia. V 6 kapitole popisuji dosažený výsledek, následuje použitá literatura a seznam příloh.

Od praxe jsem očekával rozšíření svých znalostí a především nabytí nových zkušeností z reálného pracovního nasazení. Práce v týmu a komunikace s kolegy je jen jednou z mnoha výhod a povinností, které k individuální odborné praxi patří. Mezi další bych zařadil například určitou zodpovědnost za svou práci nebo také vyšší motivaci vzhledem k možnému, ale ne jistému finančnímu ohodnocení. Velkým přínosem individuální odborné praxe je bezesporu příležitost prosadit se a zaujmout případného budoucího zaměstnavatele.

2 Poski.com s.r.o.

Obrázek 1 – Logo společnosti Poski.com s.r.o.

2.1 Seznámení se společností

Společnost Poski.com s.r.o se zaměřuje především na webdesign, webhosting, CRM a CMS systémy, internetové aplikace, reklamní kampaně, e-business a další služby internetu. Vznikla v roce 2004, ale úplný počátek historie společnosti lze datovat do roku 1998. Tomáš Posker, který dnes patří mezi spoluzakladatele Poski.com s.r.o., tehdy založil malou společnost, která se zabývala tvorbou webových stránek. Ke konci roku 1999 už byl nucen zaměstnat první pomocné síly a v roce 2003 došlo ke sloučení s podobnou společností a vznikl PA Holding. O rok později probíhá transformace na Poski.com a rok 2007 přinesl přechod na právnickou osobu Poski.com s.r.o.

Mezi zákazníky společnosti patří například WHIRPOOL SLOVAKIA spol. s r.o., Whirpool CR, spol. s r.o., Aquina, s.r.o., OZO Ostrava s.r.o., Sara Lee Czech Republic s.r.o. (Pickwick) a mnoho dalších známých společností. Poski.com se zabývá také tvorbou e-shopů, jako jsou Žaluzie24.eu, Mix-Tee, Droherie TETA Praha a další. Mezi další velkou skupinu zákazníků společnosti patří mnoho realitních kanceláří, které používají realitní systém Poski REAL, například EVROPA realitní kancelář, s.r.o., Reality Kocourek, CERET Reality, Reality World s.r.o. a další.

2.2 Popis mého pracovního zařazení

Na praxi jsem nastoupil na počátku listopadu 2012 společně s dalším spolužákem, který ale záhy odešel a na praxi jsem zůstal sám. Byl jsem zařazen na pozici Junior Programátor a začal jsem už zcela sám pracovat na projektu Poski Kecálek.

2.3 Průběh praxe

V tabulce uvádím dny strávené nad jednotlivými fázemi vývoje programu Poski Kecálek. Každá z těchto fází v sobě zahrnuje studium dokumentací, konzultace se zaměstnanci společnosti, implementaci dané fáze a případné další úpravy.

Tabulka 1 – Časový rozpis praxe

Vývojová fáze	**Počet dní**
Databáze	9
Jádro systému	22
Uživatelské rozhraní	8
Testování	6
Zkušební provoz	5

Z tabulky je zřejmé, že jsem nejvíce času strávil nad jádrem systému, které je napsáno v PHP a představuje vrstvu mezi daty a uživatelským rozhraním. Tento čas se prodloužil díky pozitivním změnám provedených v databázi, protože bylo nutno patřičně upravit i třídy a metody, které do databáze přistupovaly. Díky mé dobré znalosti webových technologií mi uživatelské prostředí nezabralo příliš mnoho času. Testování na firemních serverech a zkušební implementace Poski Kecálka na vlastní web společnosti se zdrželo z důvodu rozdílné konfigurace serverů.

3 Poski Kecálek

Má praxe ve společnosti Poski.com s.r.o. byla založena na projektu Poski Kecálek, zkráceně Kecálek, který má nahradit konkurenční řešení LiveChatoo [8]. Je to nástroj pro přímou komunikaci se zákazníky prostřednictvím vloženého chatovacího okna na webových stránkách klienta, dále také nazývaného jako plugin. Program měl umožnit snadnou a pohodlnou komunikaci s operátory s důrazem na vlastní značku „Kecálek“. Nešlo tedy o jednorázový projekt vyvíjený účelově pro jeden konkrétní web, ale o komplexní a udržovaný systém schopný spolupracovat s jakýmkoli webem. V zadání projektu se proto nacházely požadavky na uživatelsky nastavitelný vzhled a přizpůsobení designu pro daného zákazníka a jeho firemní image. Uživatelské prostředí muselo být příjemné a moderní. Dále byla nutná podpora přihlášení více operátorů najednou, stejně jako možnost multi-chatu, tedy komunikace jednoho operátora s více klienty současně. Důležitým prvkem byla i audiovizuální notifikace nových zpráv, aby operátor nemusel manuálně hlídat příchozí klienty.

Důraz byl kladen i na zabezpečení všech operací, hesel a logů. Mezi okrajové, ale neméně důležité vlastnosti a funkce patří základní statistiky, archiv chatů a panel s poznámkami, který umožňoval operátorovi zapsat a uložit důležité informace od klienta.

Abych mohl začít pracovat na zadaném projektu a realizovat své nápady, bylo nejprve nutné prozkoumat a nastudovat zdrojové kódy projektů, které budou využívat nebo se jinak podílet na mém projektu. Pro zachování know-how společnosti jsem podepsal smlouvu o ochraně obchodního tajemství. Tímto krokem jsem získal přístup do konkrétních repozitářů společnosti, kde se nacházejí veškeré zdrojové kódy potřebné pro dokončení projektu, na kterém jsem měl pracovat.

Ve společnosti je pro tento účel zaveden open source systém SVN, který je vyvíjen pod křídly Apache Software Foundation. Je to systém pro správu a verzování zdrojových kódů. Rozhodl jsem se použít vývojové prostředí Eclipse s pluginem Subversive, který integruje systém SVN přímo do IDE, protože umí výborně pracovat s nejrozšířenějšími programovacími jazyky a možnost instalace plug-inů dále rozšiřuje jeho možnosti. K výběru přispěl i fakt, že se jedná o open source software, který je zdarma a lze ho používat na všech majoritních operačních systémech i pro komerční účely. Seznámení se se zavedeným a fungujícím systémem společnosti bylo důležitou a časově náročnou etapou, kterou se rozhodně nevyplatí podceňovat.

3.1 Použité technologie a standardy

Společnost Poski.com s.r.o. využívá pro své produkty objektově orientovaný interpretovaný skriptovací jazyk PHP [2]. Je to výkonný a rozšířený jazyk, který umí výborně spolupracovat s databází MySQL, taktéž využívanou ve společnosti [1]. Jeho syntaxe je velmi podobná C a nabízí širokou paletu integrovaných funkcí včetně hashovacích a šifrovacích metod, práci se soubory, asociativní pole vhodná pro interpretaci JSON datových struktur a XML parser. Už jsem v PHP pracoval dříve a jádro projektu představující servisní vrstvu nad databází jsem se tedy rozhodl napsat právě v něm.

MySQL databáze spolupracuje velmi dobře s rozšířením PHP modulem mysqli [2]. V repositářích společnosti se MySQL běžně používá a s touto technologií jsem velmi dobře

obeznámen, proto jsem rozhodl ji použít pro návrh databázové struktury. Výhodou MySQL je také otevřený zdrojový kód [1].

Data ze servisní vrstvy napsané v PHP bylo třeba nějakým rozumným způsobem zprostředkovat uživateli v prohlížeči. Dle požadavků v zadání projektu jsem se přiklonil k nastupujícímu novému standardu HTML5 ve spojení s novými kaskádovými styly CSS3 [3, 4]. Přestože ani jeden z uvedených standardů není dosud dostupný ve své finální specifikaci a žádný z aktuálních prohlížečů nemá jeho plnou podporu, je HTML5 a CSS3 podporováno majoritními prohlížeči dle jejich aktuálního vývojového stádia [3, 4]. Zvolil jsem tyto dvě technologie nejen pro jejich inovativní a moderní funkce, ale také pro dobrou odezvu zákazníků na tyto nové standardy.

Základní vlastnosti projektu se mi podařilo vyřešit již na straně serveru, některé jsem ale musel řešit na klientské straně. K tomuto účelu jsem použil scriptovací jazyk JavaScript [6]. Jedná se o velmi rozšířený dialekt standardizovaného skriptovacího jazyka ECMA Script, který umožňuje běh klientských skriptů přímo v prohlížeči. K posílání i přijímání dat jsem se rozhodl použít technologii AJAX, která zastřešuje asynchronní požadavky řízené JavaScriptem. Tímto způsobem lze bez opětovného načtení stránky odesílat i přijímat data a vytvořit tak dojem skutečného desktopového programu.

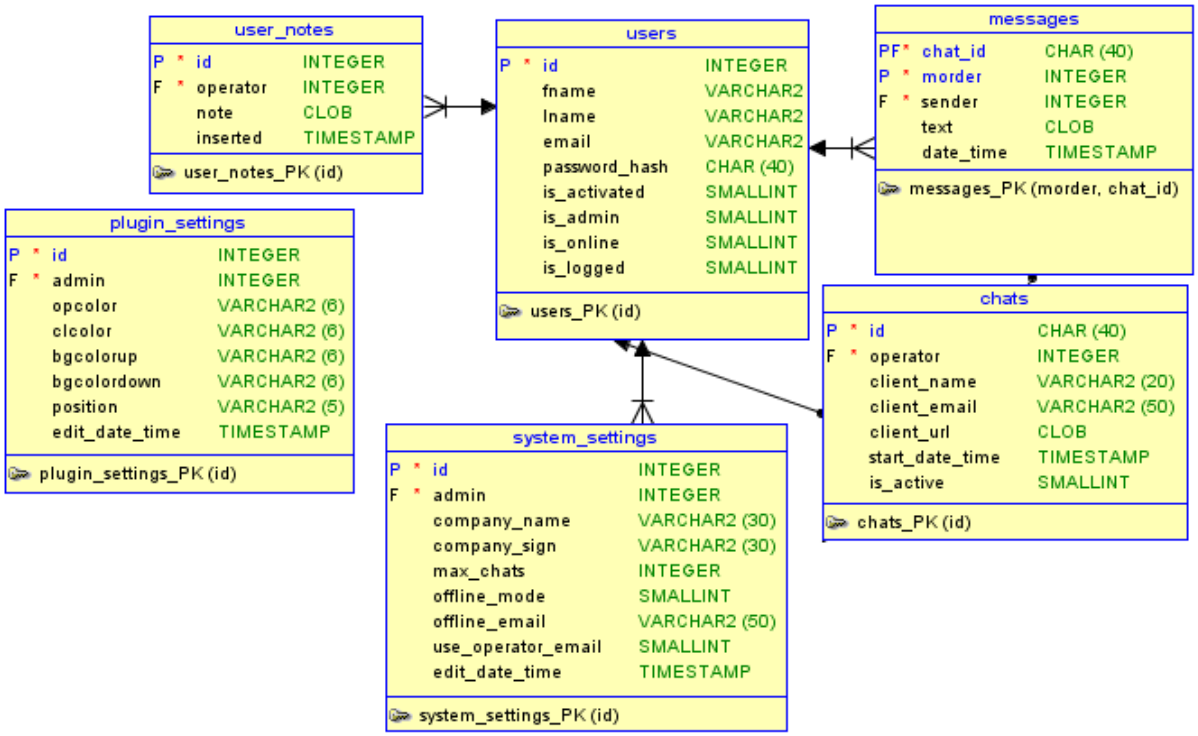
Ve společnosti Poski.com se téměř ve všech produktech využívá JavaScriptový Framework jQuery [5]. Tento Framework poskytuje sadu metod pro snadnou práci s DOM elementy, AJAX požadavky, událostmi a mnoho dalších funkcí. Poskytuje především velmi jednoduchý způsob výběru DOM elementů založený na selektorech CSS. Výsledkem použití jQuery je čistší a přehlednější kód, kompatibilita mezi prohlížeči a rychlejší vývoj.

Pro samotný asynchronní přenos dat se nabízí otevřený formát XML, který je určen právě pro použití v AJAX přenosech. XML klade náročnější požadavky na výkon klientského počítače a vhodný pro velké množství requestů za sekundu. Mnohem rychlejší, ale stále human-readable formát dat poskytuje technologie JSON [7]. Jeho syntaxe je velmi podobná konvenci jazyků rodiny C, do které patří i JavaScript. Skládá se ze dvou základních struktur a to asociativní pole nebo také kolekce párů název / hodnota a tříděným seznamem hodnot, který představuje tříděné pole nebo posloupnost. Technologicky není nijak omezen způsob zápisu dat posílaných pomocí AJAX požadavku, XML je pouze upřednostňovaný formát. JSON lze tedy bez problémů přenášet.

4 Zvolený postup řešení

4.1 Databáze

Prvním a nejdůležitějším krokem po výběru vhodných technologií byla analýza požadavků, která zjednodušila návrh databázové struktury s ohledem na možné budoucí editace a technologii MySQL [1]. Databázi jsem navrhnul Oracle SQL Developer Data Modeler. V průběhu praxe jsem databázi několikrát měnil, aby co nejvíce odpovídala požadavkům v zadání. Finální schéma databáze je na obrázku 2.



Obrázek 2 – ER diagram

První na řadě byla tabulka sloužící k ukládání a evidenci uživatelů nazvaná jednoduše users. Jednotlivé záznamy v tabulce jsou identifikovány unikátním číselným klíčem. Tento primární klíč má název id a je nastaven na automatickou inkrementaci. S každým novým uživatelem se tedy automaticky zvýší jeho id na hodnotu o jednu vyšší, než je počet záznamů v tabulce. Další sloupce obsahují základní informace o uživateli. Patří mezi ně sloupce `fname` a `lname` pro ukládání jména, sloupec email, který má příznak UNIQUE, dále pak `password_hash`, který obsahuje zašifrované heslo a nakonec následují sloupce, které mohou nabývat pouze hodnot 0 nebo 1. To je dáno integritním omezením datového typu `tinyint(1)` a tyto sloupce jsou určeny k jednoznačnému určení vlastností uživatele. Tito uživatelé se dělí na operátory a administrátory. Vzhledem k malému celkovému počtu aktivních uživatelů jsem zvolil jedinou tabulku pro oba typy entit. Tyto dvě entity rozděluje v tabulace sloupec `is_admin`. Sloupec `is_activated` určuje, zda je uživatel aktivován, neboli zda se může přihlásit do systému. Jednotlivé účty nelze z důvodu zachování integrity databáze smazat, lze je pouze deaktivovat. Deaktivovaný uživatel nemá možnost se jakkoli přihlásit do systému. Hodnota `is_logged` slouží k prevenci vícenásobného přihlášení. Poslední sloupec `is_online` má

význam pouze u entity operátora. Pomocí této hodnoty rozlišujeme, zda je operátor připraven komunikovat s klienty.

Tabulka 2 - users

název	**datový typ**	**další omezení (vyjma NOT NULL)**
id	int(11)	AUTO_INCREMENT
fname	varchar(20)	-
lname	varchar(30)	-
email	varchar(30)	-
password_hash	char(40)	-
is_activated	tinyint(1)	DEFAULT ‘0’
is_admin	tinyint(1)	DEFAULT ‘0’
is_online	tinyint(1)	DEFAULT ‘0’
is_logged	tinyint(1)	DEFAULT ‘0’

Komunikace jednotlivých operátorů s klienty musí být jednoznačně identifikovatelná a unikátní v celém systému. K tomuto účelu slouží tabulka `chats`, ve které každý řádek představuje jednu instanci komunikace. Primárním klíčem tabulky řetězec o délce 40 znaků, který musí být unikátní. Pro identifikaci operátora je připraven cizí klíč `operator`, který odkazuje na tabulku `users`. Sloupce `client_name`, `client_email`, `client_url` uchovávají informace o klientovi, konkrétně jeho jméno, e-mail a URL adresu, na které se aktuálně v rámci hierarchie webu nachází. Hodnota `start_date_time` obsahuje časový údaj o začátku konverzace a sloupec `is_active` určuje, zda je chat stále otevřen pro komunikaci, nebo zda byl již uzavřen. Uzavřený chat nelze editovat ani posílat v něm nové zprávy.

Tabulka 3 - chats

název	**datový typ**	**další omezení (vyjma NOT NULL)**
id	char(40)	-
operator	Int(11)	DEFAULT ‘1’
client_name	varchar(20)	DEFAULT ‘Client’
client_email	varchar(50)	DEFAULT ‘client@client.com’
client_url	text	-
start_date_time	timestamp	DEFAULT CURRENT_TIMESTAMP
is_active	tinyint(1)	NOT NULL DEFAULT ‘0’

Záznamy v tabulce `chats` pouze popisují jednotlivé instance chatů. Zprávy ke každému takovému chatu jsou ukládány v tabulce `messages`. Struktura tabulky sestává z cizího klíče `chat_id`, který odkazuje na identifikátor chatu z tabulky `chats`, sloupce `morder` udávajícího pořadí zaslané zprávy, druhým cizím klíčem je `sender`, který odkazuje na tabulku `users`. Samotný text zprávy je uložen ve sloupci text. Hodnota `date_time` ukládá datum a čas zaslané zprávy. Primárním klíčem této tabulky jsou sloupce `chat_id` a `morder`, pomocí kterých lze jednoznačně identifikovat záznamy.

Tabulka 4 - messages

název	**datový typ**	**další omezení (vyjma NOT NULL)**
chat_id	char(40)	-
morder	int(11)	-
sender	int(11)	-
text	text	-
date_time	timestamp	DEFAULT CURRENT_TIMESTAMP

Operátoři při komunikaci s klienty potřebují ukládat informace na zvláštní místo, aby je měli dostupné bez většího hledání v archivu zpráv pro pozdější použití. K tomu slouží jednoduché poznámky, které jsou uloženy v tabulce `user_notes`. Tabulka obsahuje pouze čtyři sloupce. První, nazvaný id, slouží jako primární klíč. Druhý sloupec `operator` je cizí klíč, odkazující na tabulku `users`. Přiřazuje danou poznámku ke konkrétnímu uživateli. Předposlední sloupec `note` obsahuje samotný text poznámky. Poslední sloupec `inserted` uchovává datum a čas vytvoření poznámky.

Tabulka 5 - user_notes

název	**datový typ**	**další omezení (vyjma NOT NULL)**
id	int(11)	AUTO_INCREMENT
operator	int(11)	-
note	text	-
inserted	timestamp	DEFAULT CURRENT_TIMESTAMP

Snadnou správu systémového nastavení zajišťuje tabulka `system_settings`. I přesto, že se nastavení skládá pouze z několika hodnot, jsem zvolil tabulku. Důvodů je hned několik. Platné nastavení je vždy to naposledy nastavené. Tuto informaci systém snadno zjistí ze sloupce `edit_date_time`, který při každém novém záznamu automaticky vkládá aktuální datum a čas. Dále lze identifikovat, kdo nastavení upravil. ID administrátora se ukládá do sloupce `admin`, který je cizím klíčem a odkazuje na tabulku `users`. Každé nastavení lze tedy zpětně dohledat, porovnat provedené změny nebo navrátit původní nastavení. Primárním klíčem celé tabulky je sloupec id. Ostatní sloupce uvedené v tabulce níže už obsahují samotné hodnoty nastavení. Aktuálně jsou

používané jen některé z nich, zbylé hodnoty naleznou uplatnění v novějších verzích. Mezi momentálně využívané hodnoty patří sloupec `max_chats`, který určuje maximální počet instancí `chatů` na jednoho operátora, dále pak `offline_mode` a související `offline_email`. První nastavuje viditelnost pluginu na webové stránce, druhý pak určuje e-mailovou adresu, na kterou se budou odesílat zprávy v případě, že je `offline` mód zapnutý a žádný operátor není zrovna k dispozici.

Tabulka.3.1: system_settings

název	**datový typ**	**další omezení (vyjma NOT NULL)**
id	int(11)	AUTO_INCREMENT
admin	int(11)	-
company_name	varchar(30)	DEFAULT 'Unknown'
company_sign	varchar(30)	DEFAULT 'Sign'
max_chats	int(11)	DEFAULT '3'
offline_mode	tinyint(1)	DEFAULT '0'
offline_email	varchar(50)	DEFAULT 'uknown@unknown.com'
use_operator_email	tinyint(1)	DEFAULT '0'
edit_date_time	timestamp	DEFAULT CURRENT_TIMESTAMP

Poslední tabulkou databázového systému je `plugin_settings`. Její struktura je podobná tabulce `system_settings`, lze tedy zpětně porovnávat, dohledávat a identifikovat všechny provedené změny. Popíši tedy pouze sloupce, které mají vliv na vzhled pluginu. Sloupce `opcolor` a `clcolor` udávají barvy v HEX formátu, kterými se budou odlišovat v chatu zprávy od operátora a od klienta. Sloupec oph3color udává barvu nadpisu pluginu, hodnoty `bgcolorup` a `bgcolordown` určují vrchní a spodní barvu pozadí pluginu, které je vyvedeno jako plynulý přechod. Poslední sloupec `position` slouží k odsazení pluginu od pravého nebo levého okraje obrazovky.

Tabulka 6 - plugin_settings

název	datový typ	další omezení (vyjma NOT NULL)
id	int(11)	AUTO_INCREMENT
admin	int(11)	-
opcolor	varchar(6)	-
clcolor	varchar(6)	-
oph3color	varchar(6)	-
bgcolorup	varchar(6)	-
bgcolordown	varchar(6)	-
position	varchar(5)	-
edit_date_time	timestamp	DEFAULT CURRENT_TIMESTAMP

4.2 Jádro systému

Jádro systému jsem si rozvrhnul do několika souborů. Každý soubor obsahuje minimálně jednu hlavní třídu. Název takového souboru obsahuje název hlavní třídy zakončený slovem Class s koncovkou php. K této hlavní třídě může být přidána další, pomocná třída, například jednoduchá třída sloužící jako kontejner pro snadné uchovávání dat. Momentálně je těchto souborů 10, celkem obsahují na 11 tříd včetně pomocných. Jednotlivé třídy postupně rozeberu, popíšu jejich funkci a uvedu vybrané důležité metody. V seznamu metod nejsou uvedeny konstruktory, destruktory a primitivní get/set metody.

DBHelper

K návrhu této třídy jsem využil vzoru Singleton. Třída obsahuje sadu metod, které usnadňují připojení k databázi a chrání databázový server před přehlcením. Díky použitému vzoru si udržuje instanci připojení k serveru a nevytváří pro každý požadavek novou instanci a tím zmenšuje zatížení serveru. Při instantní komunikaci je takových požadavků zasíláno obrovské množství.

- `public static connectToDB`

  Metoda vrací instanci připojení k serveru MySQL. Pokud není instance vytvořena, vyvolá privátní konstruktor. Vrácená třída poskytuje také metodu `real_escape_string`, která odstraňuje nebezpečné znaky z řetězců, které dosazuji do SQL dotazů [1]. Kdybych tyto nebezpečné znaky v řetězcích ponechal, znamenalo by to bezpečnostní riziko a zranitelnost systému proti útokům typu SQL injection. Metoda je bez parametrů.

- `public static query`

  Metoda přijímá jako argumenty instanci připojení k serveru, kterou lze získat pomocí metody `connectToDB`, a SQL dotaz. Vrací výsledek dotazu.

Logger

Třída Logger slouží jednoduchému ukládání logů do souboru. Soubor se vytváří pro každý den do složky logs a jeho název má tvar d-m-Y.txt. V konstruktoru předáváme jako jediný parametr úroveň ukládaných logů. Tato úroveň je reprezentována konstantami, které třída obsahuje. Všechny níže uvedené metody přijímají jediný argument, kterým je zpráva, kterou chceme zalogovat.

- `public LogError`
- `public LogWarning`
- `public LogInfo`
- `public LogDebug`

Chat

Slouží jako kontejner, který uchovává informace o chatu a všechny poslané zprávy. K tomu využívá pomocnou třídu `Message`, která obsahuje pouze privátní proměnné a get/set metody. Jednotlivé zprávy jsou ve formě instancí třídy `Message` uloženy v privátní kolekci hlavní třídy Chat.

- `public fillMessages`

  Metoda naplňuje privátní kolekci zprávami chatu. Volá se pouze v konstruktoru a při výjimečných operacích. Metoda je bez argumentů.

- `public activate/deactivate`

  Aktivuje nebo deaktivuje chat. Deaktivovaný chat neumožňuje nadále zasílat jakékoli zprávy. Metoda je bez argumentů.

- `public sendMessage`

  Pomocí této metody můžeme zaslat zprávu a přidat ji tak do kolekce zpráv. Metoda přijímá dva argumenty, prvním je samotná zpráva, druhým identifikujeme odesílatele.

- `public commit`

  Tato metoda slouží k uložení veškerých provedených změn atributů do databáze. Nevolá se při zasílání zprávy. Metoda je bez argumentů.

ChatService

Je to vrstva, která se stará o dolování dat z databáze a vytváření instancí třídy `Chat`. Kontroluje, zda požadovaná instance `Chatu` existuje a vytváří instance nové.

- `public static getActiveChats`

  Vrací všechny aktivní chaty, které moderátor v danou chvíli má. Jako parametr přijímá ID operátora.

- `public static getChat`

Vrací určitý chat dle jeho ID. ID chatu přijímá jako jediný parametr. Tato metoda se vyžívá převážně ve veřejném API, kde si klient drží pouze ID svého chatu.

- `public static createNewChat`

  Vygeneruje nové náhodné ID a zavolá konstruktor chatu. Metoda vrací instanci vytvořeného chatu. Metoda je bez argumentů.

Note

Tato třída slouží pouze jako kontejner pro poznámky. Jedna instance se rovná jedné poznámce. Držet poznámky v kolekci není potřeba, protože se s poznámkami nikdy neprovádějí žádné hromadné operace.

NoteService

Podobně jako třída `ChatService` slouží jako prostředník mezi databází a systémem.

- `public static addNewNote`

  Metoda vloží do databáze novou poznámku. Jako parametr přijímá ID operátora a text poznámky.

- `public static removeNote`

  Metoda smaže poznámku s určitým ID, které je předáváno jako parametr metody.

- `public static getNotes`

  Vrací kolekci instancí všech poznámek operátora. ID operátora je předáno jako argument.

PluginSettings

Tato třída umožňuje získat nastavený kaskádový styl pluginu. Má privátní konstruktor a její instanci lze získat voláním statické metody `getPluginSettings`.

- `public static getPluginSettings`

  Získává nastavení z databáze a vrací instanci této třídy. Metoda je bez argumentů.

- `public commit`

  Ukládá do databáze nastavení uložené v instanci třídy. Metoda je bez argumentů.

SystemSettings

Má naprosto stejnou konstrukci jako třída `PluginSettings`. Opět využívá privátního konstruktoru a obsahuje metodu `commit`. Statická metoda pro získání instance má v tomto případě název `getPluginSettings`.

User

Tato třída slouží jako kontejner pro uživatelská data. Mimo to obsahuje také několik metod pro práci s daným uživatelem. Instanci této třídy lze získat z bezpečnostních důvodů pouze z přidružené servisní třídy.

- `public online/offline`

  Metoda slouží nastavení statusu uživatele. Pokud je on-line, je zobrazen klientům k dispozici. Metoda je bez argumentů.

- `public login/logout`

  Nastaví statut uživatele. Pokud je uživatel označen metodou `login`, nelze se znovu přihlásit do stejného uživatelského účtu, dokud se původní přihlášená instance nezruší metodou `logout`. Metoda je bez argumentů.

- `public activate/deactivate`

  Pokud je uživatel metodou `deactivate` deaktivován, nelze se k jeho účtu přihlásit do opětovné aktivace, je automaticky odhlášen a jeho status je nastaven na off-line. Metoda je bez argumentů.

- `public changePassword`

  Speciální metoda pro změnu hesla. Heslo je v databázi uloženo v zašifrované podobě. Z bezpečnostních důvodů se toto zašifrované heslo neukládá do instance této třídy, proto je pro jeho změnu třeba speciální metoda, metoda `commit` totiž heslo nemění. Metoda přijímá jako argument nové heslo.

- `public commit`

  Metoda ukládá nastavení uživatele do databáze. Neukládá ani nemění heslo uživatele. Metoda je bez argumentů.

UserService

Speciální servisní třída, která zajišťuje ověření a přihlášení uživatele. Obsahuje get metody pro získání jak jednoho konkrétního uživatele, tak všech uživatelů a také dostupných operátorů, kteří jsou on-line.

- `public static addNewUser`

  Metoda má celkem čtyři argumenty, jméno, příjmení, email a heslo uživatele, kterého chceme vytvořit. Ostatní parametry se vyplňují přednastavenými hodnotami. Vrací instanci nového uživatele.

- `publi static proveUser`

  Metoda na základě argumentů ověří, zda se přihlašovací údaje shodují s údaji v databázi. V případě úspěchu zavolá metodu `login` na instanci daného uživatele a tuto instanci vrátí.

4.3 Uživatelské rozhraní operátorů a administrátorů

Jak už jsem v podkapitole použitých technologií a standardů uvedl, rozhodl jsem se pro návrh uživatelského rozhraní využít HTML5 a CSS3 spolu s JavaScriptovým frameworkem jQuery ve verzi 1.9.1 [3, 4, 5, 6]. Abych ušetřil datový prostor a zároveň zjednodušil úpravy jednotlivých stránek, rozdělil jsem strukturu stránky na tři soubory s koncovkou PHP. První soubor s názvem head.php obsahuje hlavičku, která je společná pro všechny stránky a vkládá se na začátek stránky. Jakákoli změna v hlavičkovém souboru se tedy projeví na všech stránkách najednou. První řádek v hlavičce je PHP kód, který vloží do hlavičky konfigurační soubor index.local.php a nastartuje session metodou `session_start`. Session se může také startovat automaticky, pokud je to nastaveno v konfiguraci serveru. V hlavičce je také uveden DOCTYPE, v mém případě HTML5 [3]. Dále jsou v hlavičce meta tagy, které určují kódování obsahu, titulek stránky, link na CSS styl a script tagy, které vkládají JavaScriptový kód a jQuery framework.

```
<!DOCTYPE html>
<html>
<head>
<title>Poski Kecálek</title>
<meta charset="utf-8" />
<link rel="stylesheet" type="text/css" href="style.css">
<script type="text/javascript" src="jquery-1.9.1.min.js"></script>
<script type="text/javascript">
//JavaScriptový kód
</script>
</head>
```

CSS soubor, který obsahuje kaskádové styly, jsem napsal tak, aby využíval co možná nejvíce nových vlastností CSS3. Všechny informace o CSS jsem čerpal z oficiální dokumentace W3Schools [4].

Dále hlavička obsahuje PHP kód, který na základě instance uživatele, která je uložená v session serveru, zobrazí menu pro operátora nebo pro administrátora a příslušný uživatelský panel. V případě, že session není nastavena, zobrazí menu pro nepřihlášené uživatele [2].

Obrázek 3 - Ukázka menu a uživatelského panelu operátora

Dalším souborem je patička, footer.php. Ta obsahuje pouze název společnosti a odkaz na podporu. Patičkový soubor se také vkládá do každé stránky, ale nakonec.

Třetím a nejdůležitějším souborem je obsah samotné stránky. Pro každou stránku existuje jeden takový soubor, na jeho začátku vkládám hlavičku, pak následuje tělo, neboli samotný obsah a vše zakončuje vložení patičky.

```php
<?php
include_once 'head.php';
?>
<div id="content">
//Konkrétní obsah stránky.
</div>
<?php
include_once 'footer.php';
?>
```

Uvnitř tagu div s id content se nachází obsah stránky, který je ještě zaobalen do kontrolního kódu PHP, který se stará o to, aby nepřihlášení uživatelé nebo uživatelé, pro které není obsah určen, neměli přístup k této stránce. Místo požadovaného obsahu se v takovém případě zobrazí upozornění o zamítnutí přístupu.

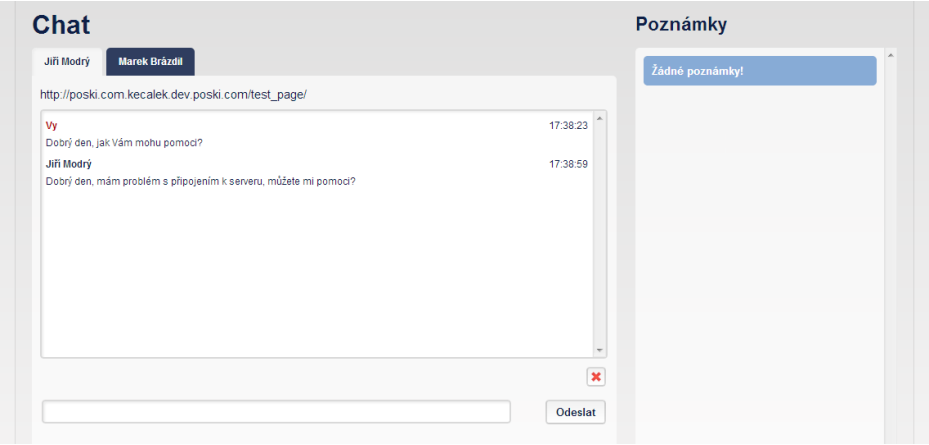
V konfiguračním souboru index.local.php jsou definovány konstanty metodou `define`. Tyto konstanty obsahují IP adresu SQL serveru, název databáze, do které budu ukládat data, uživatelské jméno a heslo k SQL serveru. Dále obsahuje inicializační metodu `init_set`, kterou mohu zapnout nebo vypnout zobrazování PHP chyb. Úplně nakonec vkládám do konfiguračního scriptu soubor classes.php, který obsahuje vložené všechny třídy a do globální proměnné nastavuje instanci třídy Logger. V konstruktoru této třídy nastavuji úroveň logování. Logger pak volám dle potřeby pomocí globální proměnné [2].

Přihlašování a odhlašování uživatelů se provádí speciálními PHP soubory, které neobsahují vloženou hlavičku a patičku. Pro přihlášení se používá soubor login.php, do kterého vložím konfigurační soubor index.local.php, nastartuji session metodou `session_start` a zavolám statickou metodu `proveUser` třídy UserService. Jako argumenty předávám přihlašovací jméno a heslo zaslané metodou POST, která je pro tento účel bezpečnější. Pokud metoda `proveUser` vrátí zpět uživatele, znamená to, že ověření proběhlo v pořádku a do session se uloží instance tohoto uživatele. Pokud se ověření z nějakého důvodu nezdaří, session se nenastaví. Nakonec dojde k přesměrování na úvodní stránku index.php. Odhlašování uživatele se provádí souborem logout.php, který má obdobnou strukturu jako soubor pro přihlašování login.php. Po nastartování session se na instanci uživatele, který je uložen v session, zavolá metoda `offline`, která mu nastaví offline status a zneviditelní ho tak pro klienty, a metoda `logout`. Následuje zrušení instance uživatele v session a přesměrování na úvodní stránku index.php [2].

Dalším speciální souborem je operator_toggle.php. Pokud je v session uložen uživatel a jedná se o operátora, přepne jeho status podle předchozího nastavení buď na online nebo offline. Po úsoěšném přepnutí script operátora přesměruje na stránku operator_chat.php.

Pro komunikaci s klienty je připravena stránka operator_chat.php. Ta vyžaduje ke své správné funkci vlastní soubor s JavaScriptem, který je uložen v souboru operator_chat.js a vkládá se do PHP souboru. Obsahuje sadu metod pro správu AJAX požadavků, generování konverzací a karet s klienty na základě přijatých dat ve formátu JSON [7]. Struktura JSON je předem známa a práce s přijatými daty je snadná a rychlá, mnohem rychlejší než s XML formátem. Pokud je DOM dokumentu načten, spustí se inicializační metoda a nastaví listenery na události klinutí myší na tlačítka. Každému listeneru přiřadí určenou metodu, která se zavolá v okamžiku, kdy uživatel klikne na daný DOM element. Dále nastaví interval stahování dat na 600ms. Každých 600ms se pak zavolá metoda

`getData`, která si vyžádá AJAX požadavkem aktuální data. Tyto data obsahují veškerou konverzaci a jména klientů, dále také obsahují URL adresu klienta, na které se aktuálně nachází. Z těchto dat se následně vygenerují zprávy a karty s klienty, které operátor vidí. Pokud se objeví nějaký nový klient, který v předchozí sadě dat nebyl, vytvořená karta se rozbliká a přehraje se notifikační zvuk, který upozorní operátora na nově příchozího klienta. Stejná situace nastává v případě, že klient zašle zprávu a operátor nemá otevřeno jeho okno. Přehrávání zvuku jsem řešil vložením HTML5 elementu audio [3]. Ukázka výřezu rozhraní operátora je na obrázku *4.1*.



Obrázek 4 - Ukázka rozhraní operátora s dvěma klienty

Operátor si může také vkládat vlastní poznámky, které se zobrazují v panelu vedle chatu, na obrázku *4.1* vpravo. O ukládání a mazání poznámek se také stará AJAX, takže nedojde k znovu načtení stránky, ale vše se děje na pozadí.

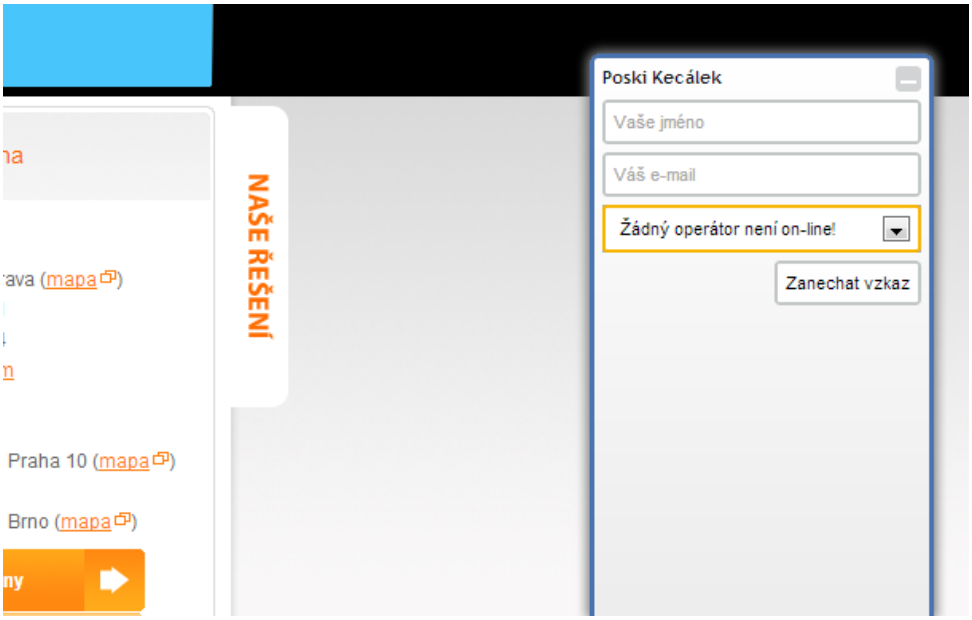
Nezbytným souborem pro správnou funkci souboru operator_chat.php je takzvané API. To je rozděleno na dvě části, a to privátní a veřejné. Pro operátora je důležitá privátní část, veřejná část je určeno pro plugin. Privátní API zpracovává veškeré AJAX požadavky zaslané JavaScriptem z chatu operátora. Všechny požadavky jsou zasílány GET metodou. Podle sady klíčů, které jsou zaslány, zvolí aplikační rozhraní odpovídající funkci, která vygeneruje požadovaná data ve formátu JSON nebo provede jinou činnost. Přítomnost klíču v GET požadavku se testuje metodou `array_key_exists`, které jako parametr předávám název klíče, který hledám, a pole, ve kterém chci klíč vyhledat. Klíče mohou být kombinovány pro provedení více akcí jedním požadavkem.

Plugin je vkládán do HTML dokumentu JavaScriptem, který stáhne HTML, CSS i JavaScriptové soubory pluginu ze serveru a zavolá metodu `init`, který plugin Kecálka vloží do DOM dokumentu a stáhne inicializační data [5, 6]. Soubory, které script stahuje, se jmenují plugin_html.php, plugin_style.css a plugin.js. Kaskádové styly pluginu využívají takové selektory, aby neovlivňovaly žádným způsobem stránku, na které je plugin zobrazen.

Veřejná část API obsahuje méně funkcí a rozeznává také méně klíčů. Její funkce jsou využívány pluginem, který je vložen do cílové webové stránky a pomocí kterého klienti komunikují s operátory.

Obrázek 5 - Zavřený plugin na stránkách společnosti Poski.com s.r.o.

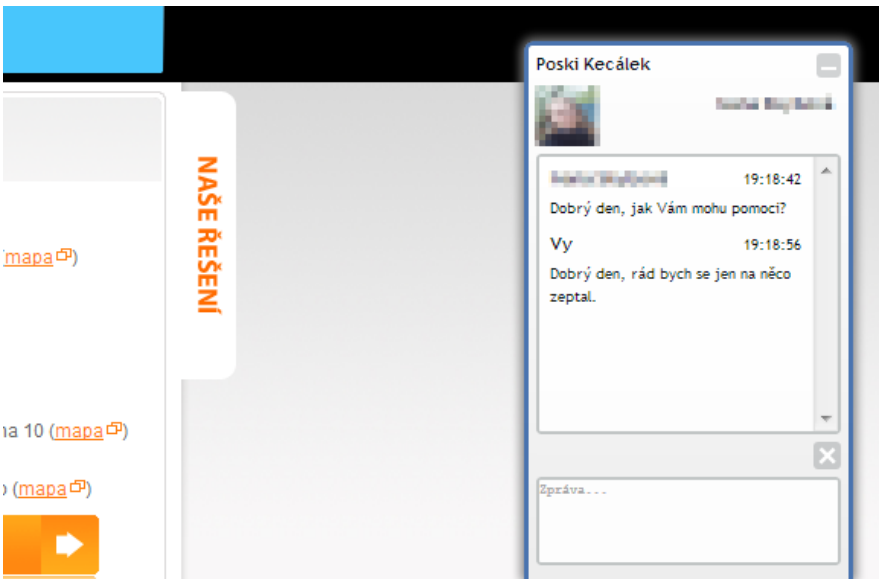
Pokud klient dosud nezačal komunikaci s žádným operátorem, zobrazí se mu zavřený plugin jako na obrázku *4.4*. V případě, že se klient rozhodne oslovit operátora, vysune stiskem maximalizačního tlačítka v pravé části pluginu nebo bubliny celý plugin v jeho plné velikosti, jako na obrázku *4.5*.



Obrázek 6 - Otevřený plugin na stránkách společnosti Poski.com s.r.o.

Klient vyplní požadované údaje a vybere si operátora z nabídky. Pokud zrovna není žádný operátor k dispozici, zobrazí se místo tlačítka „Začít chat“ tlačítko „Zanechat vzkaz“, jako na obrázku *4.5*. V případě, že je některý z operátorů dostupný a klient si jej vybere, přepne se plugin do

chatovacího režimu a komunikace může začít, viz obrázek *4.6*. Pokud klient během komunikace přejde na jinou stránku v rámci webové prezentace, na která začal komunikovat s operátorem, jeho komunikace se zachová a po načtení nové stránky se plugin otevře se všemi zprávami, které byly dosud zaslány a klient může pokračovat v započaté komunikaci. Operátor v tuto chvíli uvidí jen změnu URL adresy klienta a může tak na pohyb klienta reagovat.



Obrázek 7 - Komunikace klienta s operátorem na stránkách společnosti Poski.com s.r.o.

Kromě těchto speciálních scriptů obsahuje Kecálek spoustu dalších souborů pro různé účely. Jedná se například o přístup do archivu, statistik, nastavení uživatele, kde si může uživatel změnit heslo, jméno, příjmení a email, dále pak správu uživatelů pro administrátory, nastavení celého systému, nastavení barevného schéma pluginu, jednotlivá menu a panely, které jsou různé pro operátory a administrátory. Všechny využívají ke své funkci mnou vytvořené třídy. Součástí projektu je také několik obrázků, které jsem vytvořil v open-source grafickém editoru GIMP a jeden zvukový soubor, který slouží jako notifikační zvuk.

5 Uplatněné a chybějící znalosti a dovednosti

V průběhu praxe jsem využil znalostí nabytých nejen během studia na VŠB, ale také těch, které jsem získal samostudiem. I přesto, že jsem s HTML a CSS pracoval již dříve, získal jsem v předmětu Vývoj internetových aplikací nové znalosti, především v oblasti JavaScriptu a AJAX technologii, kterou jsem předtím neznal. Naopak formát JSON jsem si musel nastudovat sám, protože mi znalosti v této oblasti chyběly. Návrhové vzory, které jsem studoval v předmětu Softwarové inženýrství, jsem několikrát využil během praxe. Také znalosti z předmětů zabývajících se databázemi jsem během této praxe využil, když jsem navrhoval strukturu databáze. Neocenitelnými předměty, jejichž náplň jsem během praxe uplatnil, jsou Algoritmy a Programovací jazyky.

6 Závěrečné shrnutí

Absolvování této odborné praxe vnímám velmi pozitivně. Získal jsem mnoho nových a cenných zkušeností, které bych u normální bakalářské práce nezískal. Navíc jsem dostal příležitost zařadit se do týmu prosperující společnosti a podílet se tak na reálných produktech. Na projektu Poski Kecálek budu nadále ve společnosti pracovat a budu se snažit jej zlepšovat a přizpůsobovat potřebám klientů. V tuto chvíli běží Poski Kecálek na domovských stránkách společnosti Poski.com s.r.o. a připravují se další implementace.

Další oblasti možného vylepšení vidím především v lepší správě příchozích klientů, kdy si operátoři budou moci klienty předávat dle toho, jaké mají kompetence. Další užitečnou funkcí by mohlo být zasílání souborů mezi klientem a operátorem, například by šlo o různé logy, textové soubory a dokumenty. Také design a způsob výběru operátorů by mohl být automatický a více anonymní, klient nepotřebuje vědět, s kým zrovna hovoří, důležité je pro něj to, že mu některý z operátorů poskytne pomoc. Rád bych také připravil vlastní strukturu JSON dat inspirovanou protokolem http a jeho hlavičkou.

V budoucnu bych se rád dočkal většího rozšíření programu Poski Kecálek a bylo by pro mne velkým zadostiučiněním, kdyby si našel své místo na trhu on-line chatovacích programů.

7 Použitá literatura

[1] HINZ, Stefan, Paul DUBOIS, Jonathan STEPHENS, Philip OLSON a Daniel PRICE. ORACLE CORPORATION AND/OR ITS AFFILIATES. *MySQL Documentation: MySQL Reference Manuals* [online]. 5.6. 2013 [cit. 2013-05-07]. Dostupné z: http://dev.mysql.com/doc/

[2] ACHOUR, Mehdi, Friedhelm BETZ, Antony DOVGAL, Nuno LOPES, Hannes MAGNUSSON, Georg RICHTER, Damien SEGUY a Jakub VRANA. THE PHP GROUP. *PHP: PHP Manual* [online]. 5.4.14. 2013 [cit. 2013-05-07]. Dostupné z: http://php.net/manual/en/index.php

[3] HUNT, Lachlan. W3C. *HTML5 Reference: The Syntax, Vocabulary and APIs of HTML5* [online]. 2010 [cit. 2013-05-07]. Dostupné z: http://dev.w3.org/html5/html-author/

[4] W3SCHOOLS. *CSS Reference* [online]. 2013 [cit. 2013-04-28]. Dostupné z: http://www.w3schools.com/cssref/default.asp

[5] METHVIN, Dave, WORTH, Scott GONZÁLEZ, Rick WALDRON. THE JQUERY FOUNDATION. *JQuery API Documentation* [online]. 2013 [cit. 2013-04-28]. Dostupné z: http://api.jquery.com/

[6] W3SCHOOLS. *JavaScript Tutorial* [online]. 2013 [cit. 2013-04-28]. Dostupné z: http://www.w3schools.com/ajax/

[7] W3SCHOOLS. *JSON Tutorial* [online]. 2013 [cit. 2013-04-28]. Dostupné z: http://www.w3schools.com/json/

[8] NOLIMIT | DEVELOPERS, s.r.o. *Live Chat Software: Livechatoo* [online]. 2013 [cit. 2013-05-02]. Dostupné z: http://www.livechatoo.com

Přílohy

Součástí BP je CD.

Adresářová struktura přiloženého CD:

- CD obsahuje bakalářskou práci ve formátu DOCX PDF/A. Soubory jsou uloženy v kořenovém adresáři pod názvy souborů mar0127_bp.docx a mar0127_bp.pdf.
- CD obsahuje ER diagram databáze v kořenovém adresáři pod názvem souboru er_diagram.png.
- CD obsahuje screenshoty hotového produktu ve složce Screenshots ve formátu PNG.